CLR's Role in the Security Tapestry

Charlie Kaufman
Microsoft Corporation

Agenda

- **Why this stuff is important!**
- Sandboxing in its many forms
- How the CLR works
- The hard problems that remain

CLR's Role in the Security Tapestry

Charlie Kaufman
Microsoft Corporation

Agenda

- **Why this stuff is important!**
- Sandboxing in its many forms
- How the CLR works
- The hard problems that remain

Is this asking so much?

- I'd like to be able to trade stocks over the web with security comparable to calling my broker on an 800 number
- (Note: this is not a very high bar… there is a lot that can go wrong over the phone)

Things I would expect of a telephone

- Only someone really motivated can eavesdrop
- I know I'm talking to the party whose number I dialed
- The party I call may learn my phone number, but won't learn who else I've called or what I said to them
- I can talk to anyone I want without fear of damaging my phone

Why is this hard?

- Active Content
- If you run a program I wrote, it can do things with your rights behind your back
  - Read your private files and send them to me
  - Delete or mangle files you have rights to access
  - Send email composed by me but sent (provably!) by you
  - Authorize me to do things later (if you can add me to ACLs)

Why would you run a program I wrote?

- I might trick you into it
  - Look at this great game!
- You might not think of it as running a program
  - Display a PostScript file
  - Display a Microsoft Office document
  - Visit my web site
  - Open my email message

Why should any of these involve running a program?

- Bandwidth and storage efficiencies
  - A program can more concisely express something than a static description
- Flashy graphics and interactive animation
- Extensibility

Is there any safe way to run my program?

- Since the '50s, computers were designed to safely run programs from hostile users
  - Memory bounds checking built into hardware
  - Operating systems enforced access control lists
  - But… assumed users responsible for the software they ran
  - Users need to easily (and mostly transparently) run programs with a subset of their rights

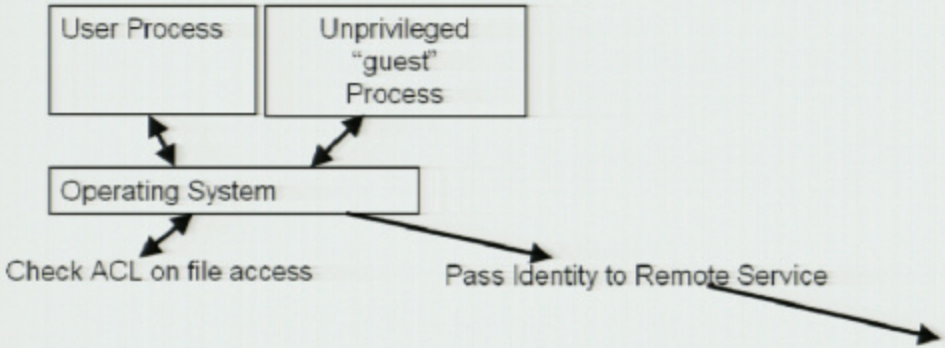
Agenda

- Why this stuff is important!
- **Sandboxing in its many forms**
- How the CLR works
- The hard problems that remain

Sandboxing

- Running a program with limited access to resources
  - Traditionally, at the level of the process
  - Alternately, you can emulate PC hardware and run an entire virtual machine
  - Computers are cheap enough that you can even use separate physical machines
  - With strong typing, secure isolation can be achieved within a process

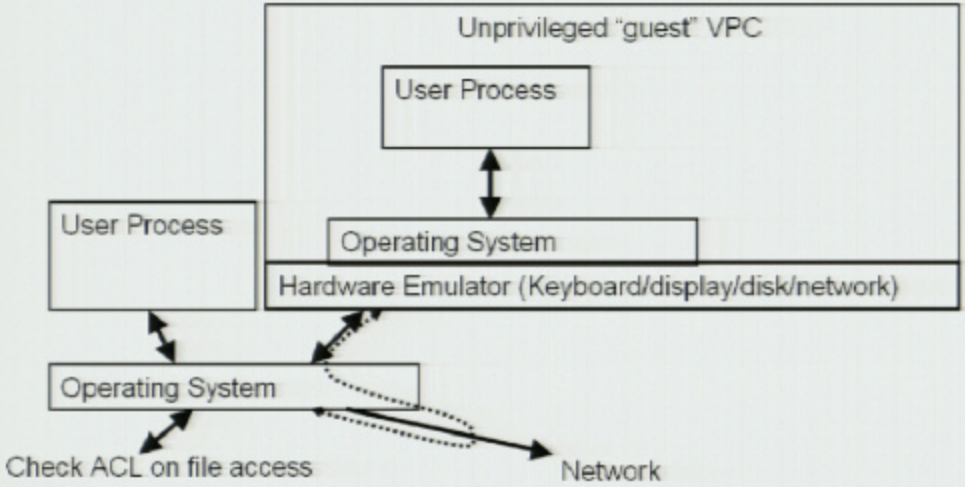
Sandboxing with Processes



Sandboxing with Processes

- A process should be able to launch a process with lesser rights
  - I want to run your game, or display your PostScript, but I don't want it to be able to open my files
- A process should be able to communicate with a process with greater rights
  - Your program should be able to access databases through a DBMS, where the DBMS knows who you are and enforces ACLs

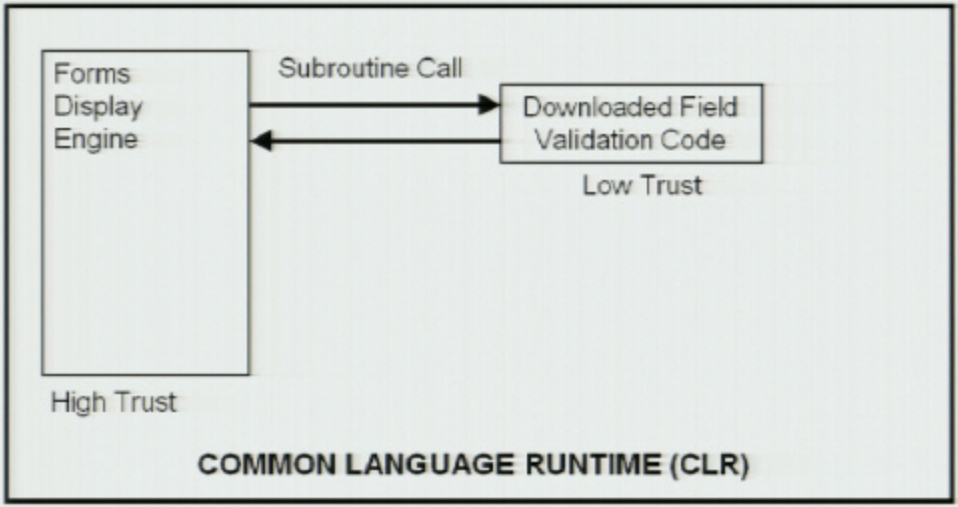
Sandboxing with VPC



Sandboxing with VPC

- A virtual PC should have no access to host system resources except through the emulated hardware
- File sharing and other interaction should use network protocols (as though there were separate physical machines)
- Keyboard/Display/Clipboard sharing represent special challenges

Sandboxing within a Process



How does strong typing help?

- In a traditional programming language, testing of data types and array bounds are an aid to help detect bugs
- With strong typing, even a malicious program cannot access process memory to which it is not given access
- Theoretically, hardware memory bounds protection becomes unnecessary (or at least redundant)

Which form of sandboxing is best?

- It depends on what you're trying to do
  - Security
  - Performance
  - Functional Richness
  - Compatibility
  - User Experience
- We plan to implement all of them for different scenarios

How secure can you make a sandbox?

- Bugs in the enforcement code can allow sandboxed code to “escape”
- Bugs in services offered can allow sandboxed code to elevate its privilege
- Bounds checking in hardware improve performance, but for security you have to have a correct TCB:
  - OS for process isolation
  - JIT, CAS, and privileged services for CLR
  - Hypervisor for VPC

How performant can you make a sandbox?

- Cost of creating and deleting sandboxes
  - How much state do they carry
- Overhead of emulation of services outside the sandbox and of context switches
- CLR better than Process better than VPC

How much functionality can you safely expose in the sandbox?

- In theory, they all could safely expose the same functionality
- In practice:
  - Because CLR limits language expressiveness, it is an unlikely host for “full function” applications
  - Processes and VPC are rich enough for all but specialized hardware interfaces

How backwards compatible can you make a sandbox

- VPC can (in theory) perfectly emulate native machine except for performance of access to specialized hardware
  - In practice, limited hardware emulation
- Process isolation can run unmodified applications that don't elevate privilege
- CLR does not attempt to sandbox applications that were not designed to fit in the sandbox

How transparent can you make sandboxes to the user?

- Transparency is not always a good thing
- The system has to figure out what the user wants
- With the CLR, application code can implement the user interaction experience (e.g. prompting for a file to open)
- With process isolation, and even more so with VPC, shared GUI and system clipboard are difficult to implement

What do we use them for?

- VPC
  - Server Consolidation
  - Compatibility testing with diverse configurations
  - Future direction: Red/Green client protection
- Process Isolation
  - Isolation of services on a shared server
  - Terminal Server
  - Unprivileged user vs. Administrator
  - Future direction: Processes with subset of user's permissions (e.g. LUA / PA)

What do we use them for?

- Code Access Security / CLR
  - Sandboxed applications launched from web sites (comparable to Java and JScript)
  - ASP.NET sandboxed servlets
  - Future direction: richer application environment for web applications
    - Goal is to support a safe rich high performance user interaction with server based applications (e.g. games, teleconferencing, brokerage sites, auction sites)

Agenda

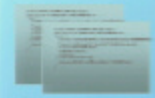
- Why this stuff is important!
- Sandboxing in its many forms
- **How the CLR works**
- The hard problems that remain

Basics of CLR

- Compilers generate 'IL' instead of native machine code
- IL is a strongly typed assembly language
- IL is 'verified' and compiled into native machine code by a Just In Time (JIT) compiler
- JIT assures type safety enforced, but produces fast machine code
- IL step makes verification possible in a language independent way and provides platform independence

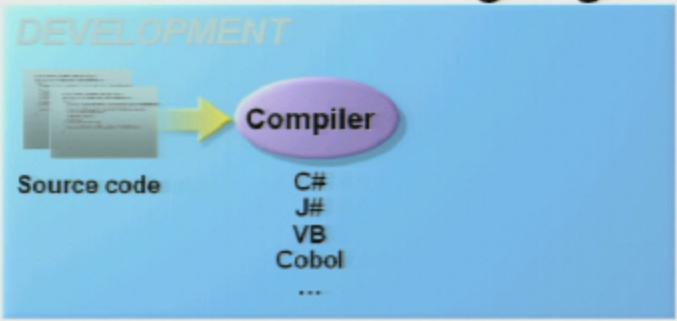
Common Language Runtime

DEVELOPMENT

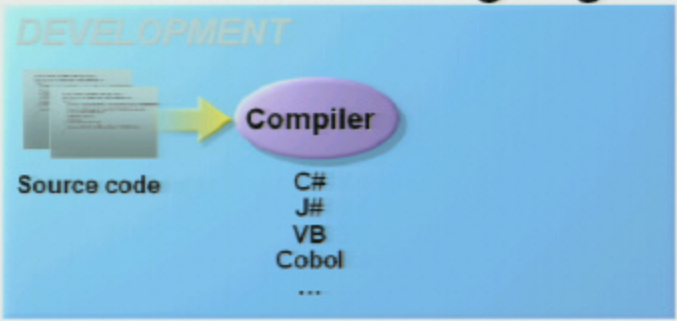


Source code

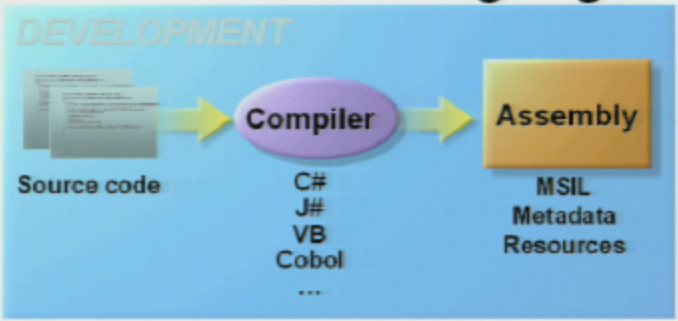
Common Language Runtime



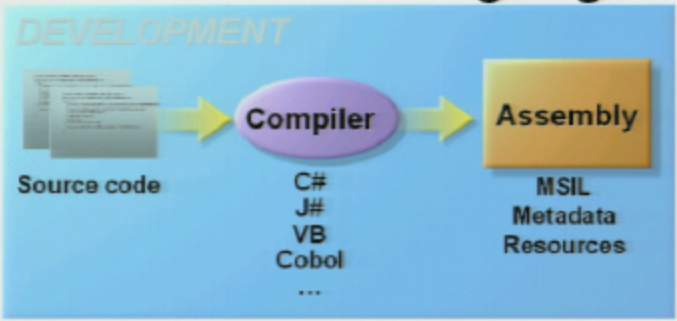
Common Language Runtime



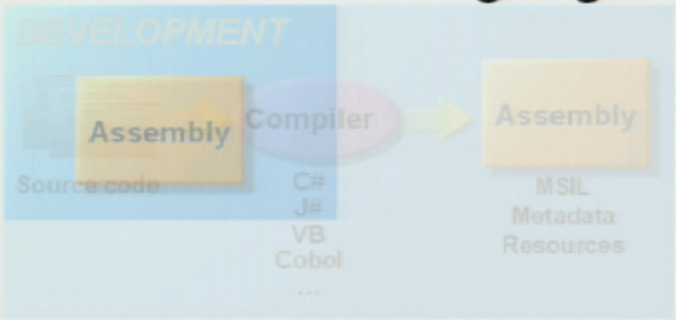
Common Language Runtime



Common Language Runtime

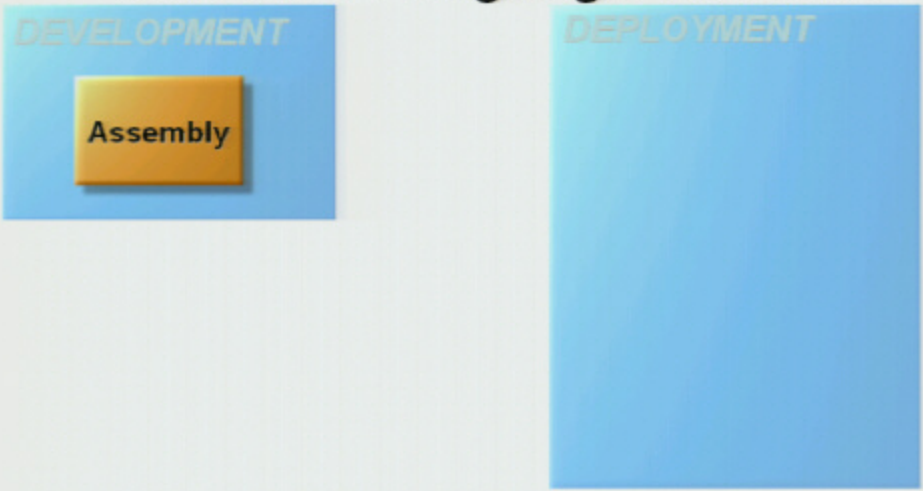


Common Language Runtime

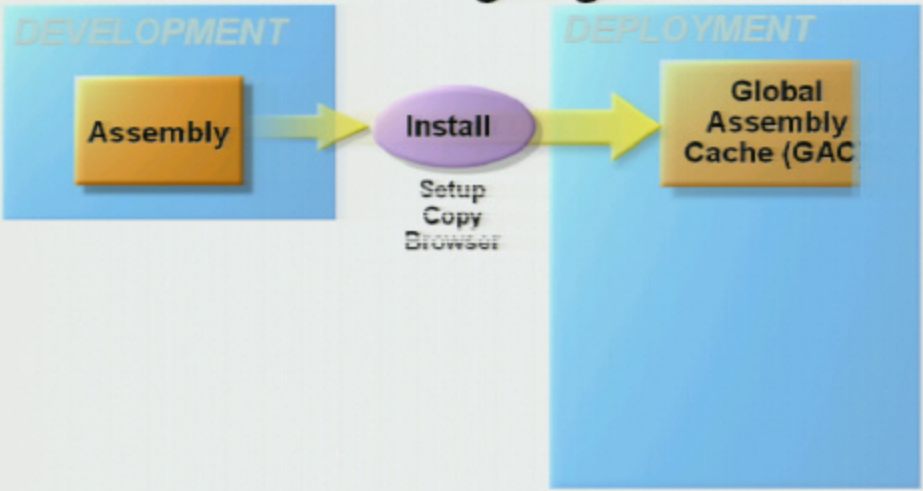


Common Language Runtime

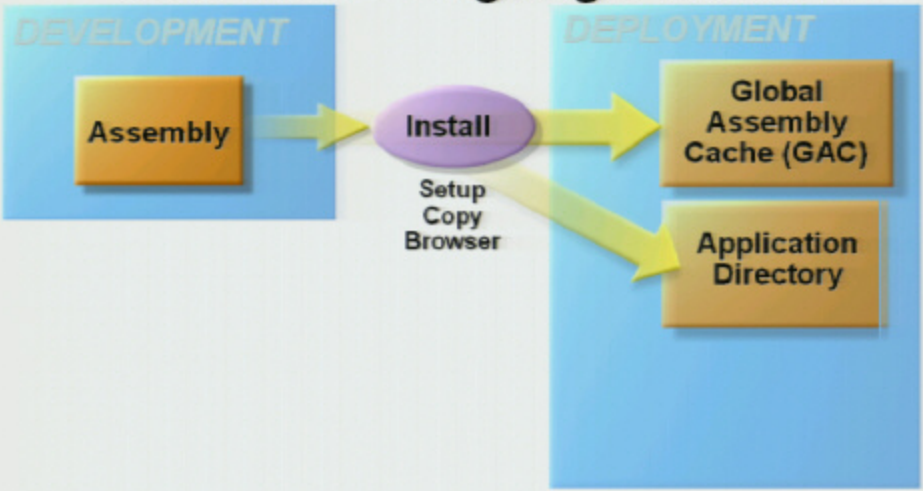
Common Language Runtime



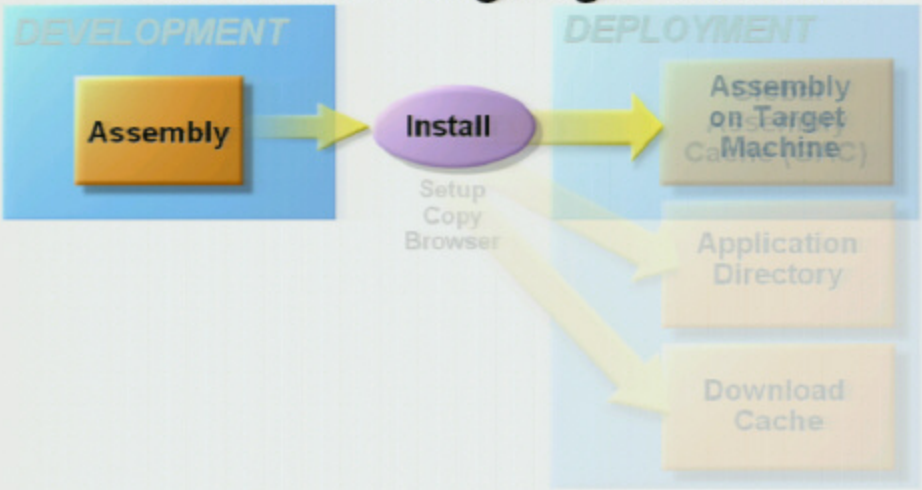
Common Language Runtime



Common Language Runtime



Common Language Runtime



Common Language Runtime

Common Language Runtime

Common Language Runtime

Agenda

- Why this stuff is important!
- Sandboxing in its many forms
- How the CLR works
- **The hard problems that remain**

Problem 1: How to decide what code to trust?

- Based on digital signatures from publishers
- Based on the web site they came from
- Based on configuration settings (the architect's cop out – how is the administrator to know)
- The code that loads the code might have an opinion
- Based on user input

CLR's Approach

- Trust Policy Engine
- Input is assembly 'evidence'
- Enterprise, Machine, User, and Application policy levels are evaluated and intersected
- Click once install lets user adjust policy

Problem 2: How to make it easy to write ‘safe’ library code

- The problem:
  - Every subroutine library (assembly / .dll) has an independently determined level of trust based on its evidence
  - By default, code runs with the intersection of the rights of itself and everyone above it on the call stack
  - Code can ASSERT up to its own rights
  - Code can PermitOnly to any smaller set of rights

Problem 2: How to make it easy to write 'safe' library code

- Code must ASSERT to provide mediated access to system resources (e.g. check a 'display this string' request to make sure it fits in the destination window)
- Code must be paranoid in checking parameters to avoid luring attacks
  - Akin to the kernel code that checks requests from user processes
  - Type safety helps, but does not solve the problem

Problem 2: How to make it easy to write ‘safe’ library code

- Challenge: create coding practices that isolate security critical code so that not everything has to be inspected so carefully
- Analogy to user mode .dlls separate from kernel mode .dlls

Problem 3: What permissions are 'safe' to grant?

- In the OS, permissions mostly centered on what files can be accessed
  - Users belong to Groups
  - Users and Groups are listed on ACLs
- A downloaded game might need:
  - 'Isolated storage' for saving its own state
  - Access to system files like fonts and screen geometry information
  - Access to the network to interact with its peers in a multi-player environment

Problem 2: How to make it easy to write ‘safe’ library code

- The problem:
  - Every subroutine library (assembly / .dll) has an independently determined level of trust based on its evidence
  - By default, code runs with the intersection of the rights of itself and everyone above it on the call stack
  - Code can ASSERT up to its own rights
  - Code can PermitOnly to any smaller set of rights

Problem 2: How to make it easy to write ‘safe’ library code

- The problem:
  - Every subroutine library (assembly / .dll) has an independently determined level of trust based on its evidence
  - By default, code runs with the intersection of the rights of itself and everyone above it on the call stack
  - Code can ASSERT up to its own rights
  - Code can PermitOnly to any smaller set of rights

Problem 2: How to make it easy to write 'safe' library code

- Code must ASSERT to provide mediated access to system resources (e.g. check a 'display this string' request to make sure it fits in the destination window)
- Code must be paranoid in checking parameters to avoid luring attacks
  - Akin to the kernel code that checks requests from user processes
  - Type safety helps, but does not solve the problem

Problem 2: How to make it easy to write ‘safe’ library code

- The problem:
  - Every subroutine library (assembly / .dll) has an independently determined level of trust based on its evidence
  - By default, code runs with the intersection of the rights of itself and everyone above it on the call stack
  - Code can ASSERT up to its own rights
  - Code can PermitOnly to any smaller set of rights

Problem 3: What permissions are 'safe' to grant?

- In the OS, permissions mostly centered on what files can be accessed
  - Users belong to Groups
  - Users and Groups are listed on ACLs
- A downloaded game might need:
  - 'Isolated storage' for saving its own state
  - Access to system files like fonts and screen geometry information
  - Access to the network to interact with its peers in a multi-player environment

Problem 3: What permissions are 'safe' to grant?

- In the OS, permissions mostly centered on what files can be accessed
  - Users belong to Groups
  - Users and Groups are listed on ACLs
- A downloaded game might need:
  - 'Isolated storage' for saving its own state
  - Access to system files like fonts and screen geometry information
  - Access to the network to interact with its peers in a multi-player environment

What Permissions are ‘safe’ to grant?

- In the CLR, permissions are organized differently than in the OS:
  - Access to specific web sites
  - Access to subtrees of the file system
  - Access to specific windows on the user’s screen
  - Mediated access to files: put up a dialog box and let the user select a file, but don’t open one directly

What Permissions are ‘safe’ to grant?

- In the CLR, permissions are organized differently than in the OS:
  - Access to specific web sites
  - Access to subtrees of the file system
  - Access to specific windows on the user’s screen
  - Mediated access to files: put up a dialog box and let the user select a file, but don’t open one directly

Where are we headed with CLR?

- A Secure Execution Environment (SEE) modeled on what script on a web page downloaded from the Internet can do
  - Mediated access to file system and printer
  - Ability to connect back to its source server
  - More privileges based on configuring trusted publishers and user prompting

Where are we headed with CLR?

- Principle of Least Privilege: run applications only with rights they declare they need in their manifests
- Sand Dunes: If a sandbox constrains a program to safe things, a sand dune makes it hard to wander too far off by accident

Problem 3: What permissions are 'safe' to grant?

- In the OS, permissions mostly centered on what files can be accessed
  - Users belong to Groups
  - Users and Groups are listed on ACLs
- A downloaded game might need:
  - 'Isolated storage' for saving its own state
  - Access to system files like fonts and screen geometry information
  - Access to the network to interact with its peers in a multi-player environment

Problem 3: What permissions are 'safe' to grant?

- In the OS, permissions mostly centered on what files can be accessed
  - Users belong to Groups
  - Users and Groups are listed on ACLs
- A downloaded game might need:
  - 'Isolated storage' for saving its own state
  - Access to system files like fonts and screen geometry information
  - Access to the network to interact with its peers in a multi-player environment

Problem 3: What permissions are 'safe' to grant?

- A downloaded tax filing program would need more:
  - Access to my tax information
- How do I let a program access some sensitive things without completely opening up my system to it?

Problem 3: What permissions are 'safe' to grant?

- In the OS, permissions mostly centered on what files can be accessed
  - Users belong to Groups
  - Users and Groups are listed on ACLs
- A downloaded game might need:
  - 'Isolated storage' for saving its own state
  - Access to system files like fonts and screen geometry information
  - Access to the network to interact with its peers in a multi-player environment

Problem 3: What permissions are 'safe' to grant?

- A downloaded tax filing program would need more:
  - Access to my tax information
- How do I let a program access some sensitive things without completely opening up my system to it?

What Permissions are ‘safe’ to grant?

- In the CLR, permissions are organized differently than in the OS:
  - Access to specific web sites
  - Access to subtrees of the file system
  - Access to specific windows on the user’s screen
  - Mediated access to files: put up a dialog box and let the user select a file, but don’t open one directly

Where are we headed with CLR?

- A Secure Execution Environment (SEE) modeled on what script on a web page downloaded from the Internet can do
  - Mediated access to file system and printer
  - Ability to connect back to its source server
  - More privileges based on configuring trusted publishers and user prompting

Where are we headed with CLR?

- Principle of Least Privilege: run applications only with rights they declare they need in their manifests
- Sand Dunes: If a sandbox constrains a program to safe things, a sand dune makes it hard to wander too far off by accident

Problem 4: The Russian Doll model of Sandboxes

- I want to use Hotmail, but don't trust it
- Hotmail wants to display email messages, but it doesn't trust them
- Someone may want to send me email with an embedded ad, but he doesn't trust the ad
- The Hotmail service may be hosted on a commercial web service provider who doesn't trust the service code.

How many different trust levels?

- Isolate individual lines of code?
- Isolate assemblies?
- Isolate App Domains?
- Isolate processes?

Problem 5: Making CLR, OS, and Web Security more similar

- All need to recognize user identity and program identity is inputs to access decisions
- Differences based on their evolution, not real needs
- People have a hard enough time learning one system; they aren't going to learn three.

Questions?

Agenda – Day One

- *8:30AM* — *Continental Breakfast and Open Discussion/Updates*
- *9:30AM* — *BREAK*
- *9:45AM* — *Longhorn Security Overview*
- *11:00* — *BREAK*
- *11:15* — *Internet Explorer Security Overview*
- *12:30PM* — *BREAK / LUNCH*
- *12:45PM* — *WORKING LUNCH - Common Language Runtime - Security Architecture Overview*
- *2:00PM* — *BREAK*
- *2:15PM* — *Microsoft Privacy Standard Update – SDL Integration*
- *3:45PM* — *ADJOURN FOR DAY – BUS TO MSCC FOR TECHFEST*
- *5:45PM* — *BUS LEAVES MSR FOR HYATT BELLEVUE*
- *7:30PM* — *Dinner – Daniel's Broiler – Bellevue*